Semester 2 Examinations 2016/2017

Exam Code(s)	4BCT, 4ECE
Exam(s)	B.Sc. in Comp Sc. & Information Technology B.E. in Elect.& Comp. Engineering
Module Code(s)	CT420
Module(s)	Real-Time Systems
Paper No.	1
Repeat Paper	N
External Examiner(s)	Dr. John Power
Internal Examiner(s)	Dr. Michael Schukat *Dr. Hugh Melvin *Mr. Yusuf Cinar *Dr. Michael Schukat

Instructions: Answer Q1 and 1 other Q from section A.
Answer 2 questions from section B.
All questions carry equal marks.

Duration	2 hours
No. of Pages	6
Discipline	Information Technology

Requirements:
MCQ
Handout
Statistical/ Log Tables
Cambridge Tables
Graph Paper
Log Graph Paper
Other Materials

Release to Library: Yes [X] No [ ]

Section A

Q1

(i) Mouth to Ear delay is a key metric of Voice over IP. Explain what it is, why it is a key metric, and outline the main contributory elements on the sender, network and receiver.

[20]

(ii) Describe both Media Independent and Media Dependent Forward Error Correction (FEC) strategies for Voice over IP. Comment on their impact in terms of delay, bandwidth utilisation, and voice quality.

[10]

(iii) DASH is very widely used for streaming multimedia. Explain briefly what it is, how it works and why it works well in today's best effort Internet.

[10]

Q2

(i) Many RTS have specific requirements for both time and timing synchronisation. Distinguish clearly between these two concepts using examples to illustrate your answer for both Hard and Soft RTS.

[10]

(ii) Distinguish between narrowband (NB), wideband (WB), and super-wideband (SWB) voice codec's, outlining the advantages/disadvantages of each.

[10]

(iii) The data below shows the output of the ntpq utility from 2 NTP clients, A and B. Based on the output, which client is most likely to deliver better Time Synchronisation? Your answer should comment on redundancy, offsets, delay, jitter, stratum level and reachability. Comment also on the data in columns 'When' , and 'Poll'.

PTO

ClientA	RefID	st	t	When	Poll	Reach	Delay	Offset	Jitter
+server 1	Server x	2	u	51	64	156	391.281	6.24	7.79
-server 2	Server y	2	u	49	64	372	353.217	10.435	1.663
*server 3	Server z	2	u	50	64	356	85.688	2.465	2.666
+server 4	.PPS.	1	u	49	64	357	66.369	1.858	2.105
ClientB	**RefID**	**st**	**t**	**When**	**Poll**	**Reach**	**Delay**	**Offset**	**Jitter**
+server 5	Server m	2	u	45	64	377	36.345	2.24	3.458
-server 6	Server n	2	u	42	64	377	313.217	3.435	1.11
*server 7	.PPS.	1	u	13	64	377	10.688	1.485	1.455
+server 8	.PPS.	1	u	22	64	377	28.456	1.818	1.345
+server 9	.GPS.	1	u	18	64	356	15.345	-2.654	2.34

[20]

Q3

(i) In the context of voice communication, distinguish between intrinsic and perceived Quality-of-Service (QoS), outlining also how they can be measured.

[10]

(ii) For Voice over IP, what do you consider as acceptable intrinsic QoS? Briefly explain your answer.

[5]

(iii) As a senior network designer for an IP communications provider, you are asked to outline a complete end-to-end infrastructural design for a potential Irish client company that wishes to implement an integrated all-IP based conferencing (voice/video) and data communication system. The company will have two large facilities, in Dublin and Galway, each with their own LAN. Your design should address the following challenges:

a. How VoIP phones chosen can maximise voice quality.
b. How QoS will be achieved across company LANs.
c. How QoS will be achieved across intermediate WAN. You can assume that your company uses DiffServ in its IP network.

[25]

PTO

Section B

Q4

(i) Briefly discuss why networked systems (where packets are routed through a subnet) are usually not considered to be hard RTS. Use diagrams to illustrate your answer.

[6]

(ii) Memory locking is an important feature of POSIX 4. Explain the code snippet below and use the 7-state process model to outline why memory locking is important for (hard) real-time systems.

```
/*Main routine */
int main(void ){
/* Lock all process down */

mlockall(MCL_CURRENT|MCL_FUTURE);

… process code

munlockall();
return 0;
}
```

[14]

What are the consequences of memory locking if being applied too generously?

[2]

(iii) Outline reasons why standard operating systems kernels are usually not deployed in (hard) RTS.

[4]

(iv) Outline a potential run-time problem of the function f() below:

```
int f(int i)
{
        int n;
        if (i == 0) {
                n = 1;
        }
        return n;
}
```

[5]

How can this problem be fixed?

[1]

Use the function f() above as an example to explain the term *fault latency*.

[8]

Q5

(i) Construct a cyclic executive (CE) schedule for the set of tasks listed in Table 1 below. Furthermore, draw the CE timeline and determine how much CPU time is left for the processing of asynchronous interrupts.

Task	**Period p msec**	**Exec Time msec**
A	25	9
B	25	6
C	50	4
D	50	5
E	100	3

Table 1

[12]

(ii) Using examples, distinguish between permanent, intermittent and transient faults.
[6]

(iii) Discuss in detail how *information redundancy* can be provided:

- on RAM / EPROM-level via a voter (as seen in the Burner Management Computer System deployed in Moneypoint)
- on data level, via the concept of the Hamming Distance *hd*.
- on secondary storage level via RAID.

In your answer make reference to the Single-Error Correcting Code (SEC) and RAID-5.

[15]

(iv) What is meant by the worst-case execution time (WCET) of program code?
[2]

(v) Discuss why WCET calculations are notoriously difficult with modern CPUs that support caches or instruction pipelines.

[5]

PTO

Q6

(i) Using the task set in Table 2 below:

- Calculate the overall *CPU utilisation U*.

[2]

- Determine the process schedule using the *RM scheduling algorithm*.

[8]

Task	Execution Time	Period
1	20	100
2	30	180
3	80	240
4	70	400

Table 2

(ii) What is meant by the *priority inversion problem* and outline how it can be solved? Use an example to illustrate your answer.

[12]

(iii) Summarise the main features of POSIX.4 signals.
Further on, outline the purpose and functionality of the function:
int sigqueue(pid_t pid, int sig, const union sigval value);

[12]

(iv) Briefly discuss the concept of time redundancy in the context of backward error recovery, as seen in the recovery block approach of software redundancy.

[6]